Lab 8 – Socket Programming

In this lab you will learn about Socket Programming in Android and Java. You will open a Socket connection and be able to send String values from the Client to the Server and vice versa.

1.1 Brief introduction to Socket programming

Before you start coding, lets get familiar with Sockets.

When you need to realize a communication between a server program and a client program in a network, is when sockets come into help.

1.2 What is Socket?

A single connection between two pieces of software (where more than two pieces of software can communicate in client/server or distributed systems, for example Web Browsers)
Sockets are methods that provide the communication mechanism between two devices using TCP (Transmission Control Protocol – allows communication between two applications. It is a two-way communication protocol, so data can be sent across both streams at the same time). A client program creates its own socket and tries to connect to that socket on the server side.
When the connection is made, the server creates its own socket object. When this is done, the server and the client will be able to communicate with each other by reading from and writing to the socket.

Link to the official documentation about Sockets in Android:
http://developer.android.com/reference/java/net/Socket.html

Short Notice: Programming sockets in Java and programming sockets in Android is pretty much the same because the source in Android is the same like Java, though they don't use the same JVM

If you are not very familiar with Java yet, you can find more facts about Java on the following link:
http://code.tutsplus.com/tutorials/learn-java-for-android-development-introduction-to-java--mobile-2604

1.3 Getting started

What you are actually supposed to do is when the **Send** button is pressed, your application should send a Text message to the server using sockets. The server then, is supposed to reply that the message is received. You can choose the port that your server program will be listening to.
Have in mind the following:
System ports 0 – 1023
User ports 1024 – 49151 (The range you are interested in)
Dynamic and/or Private ports 49152 – 65535

- **Permissions**

Android actually protects resources and data with permissions. They are represented as Strings and declared in AndroidManifest.xml.
Permissions are used to limit access to:
- User information, for example contacts
- Cost – sensitive API's, for example SMS/MMS
- System resources, for example camera

For this app you only need to define to following permission in the AndroidManifest.xml file
<uses-permission android:name="android.permission.INTERNET" ></uses-permission>

1.4 AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. An asynchronous task is defined by 3 generic types: **Params, Progress** and **Result** and 4 steps called **onPreExecute, doInBackground, onProgressUpdate** and **onPostExecute**.
More information about AsyncTask:
http://developer.android.com/reference/android/os/AsyncTask.html

- **Usage**

AsyncTask must be subclassed to be used. The subclass will override at least one method **(doInBackground(Params…))**, and most often will override a second one **(onPostExecute(Result))**

For the (android) client –(java)server connection you will need the AsyncTask in order to create the connection with the server. Three AsyncTasks will be implemented: **Sender, Receiver** and **ChatOperator**.

- **Sender**

  The Sender task sends the message through the output stream
  You will implement two methods here: **doInBackground()** and **onPostExecute()**

1. Define a String called **message;**
2. In the **doInBackground(Void… params)** get the text from the TextField in the previously defined String
3. Concatenate the results of the Temperature and the Pressure in a new String;
4. Write the created string.
   Note: The difference between a PrintStream and a PrintWriter is that, the first represents a stream of bytes, while the PrintWriter is a stream of Characters.
5. Flush the writer.

For the other method, **onPostExecute()**, having in mind that it executes after the **doInBackground()** method, you need to clear the Textfield (after the text is sent) and set the text to the TextView what the Client sent to the server.

- ## Receiver

The Receiver task continuously reads the input buffer.

1. Define a new String called **message**
2. In the **doInBackground(Void… params)** in atry+catch block
   If the bufferedReady() is ready then, in the **message** place the line that is read from bufferedReader.readline();
3. Add publishPorgress(null);
   This method is used to show any form of progress in the user interface while the background computations is still executing.
4. Since the Receiver task is supposed to continuously read the input buffer, put your code in a **while()** cycle
5. try {
   Thread.sleep(500);
   } catch (InterruptedException ie) {
   }

For the **onPostExecute()** method you need to set the text of the TextView with what the Server sent back to the client

- ## ChatOperator

1. Name the class ChatOperatot and extend the AsyncTask
2. Override the **doInBackground**() method
3. In the beginning of the class define the **CHAT_SERVER_IP** with your IP addressin order to connect to the local host with Android. You can check your ip+address using the command **ipconfig** in **cmd**.
   Besides, define:
   private Socket client;
   private PrintWriter printWriter; // this is used for writing the message to output stream
   private BufferedReader bufferedReader; // to get the client message
4. In a try – catch block, create the server socket
   client = new Socket(CHAT_SERVER_IP, 4444);
   Note: The port is randomly chosen. You can use whichever port is available within the range we talked about. The server program is continuously listening to the port 4444 in this case. When a message is arrived the server reads it and shows it on the standard output.
5. If the client is not null i.e. the server has been properly started you should initialize the printWriter and bufferedReader.

printwriter = new PrintWriter(client.getOutputStream(), true);

InputStreamReader inputStreamReader = new InputStreamReader(client.getInputStream());
bufferedReader = new BufferedReader(inputStreamReader);

If the client is null then print a proper message.

6. Catch the **UnknownHostException** and **IOException** and outprint the message System.out.println("Failed to connect server " + CHAT_SERVER_IP);
7. Finally, return **null** in the **doInBackground()** method.

Another method you should implement in the **ChatOperator** is the **onPostExecute(Void result){..}** This method is executed at the end of the **doInBackground().**
Since this methos is executed at the end of the previous methos, here you should call the button Listener i.e. here when pressed on the button **Send** a message from the client (and you already opened the connection) should be send to the server.

1. Inside the **setOnClickListener** you should initialize the **Sender** AsyncTask
2. Add the code below after the initialization

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
messageSender.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR);
    } else {
    messageSender.execute();
    }
```

3. Initialize the **Receiver** AsyncTask
4. Execute the receiver.

1.5 Java Socket Programming

For the implementation of the requirements you will create three classes in Eclipse.
Create three classes named: **Receiver, SimpleServer** and **ChatClass**

- **Receiver**

This class works as a thread and continuously will check whether a message is available in the inputbuffer. Its implementation is very similar to the one in Android.

1. Create the **PrintWriter** and **BufferedReader** in the beginning of the class
2. In the constructor of the class the argument will be the client's socket.
3. In a try – catch block initialize the **PrintWriter** and the **BufferedReader**
4. Inside the run method: (**Question:** What do you think it is inside the run method?)
   Create a string and do the same steps as for Android.
5. In the console, print the message from the client. The code inside the try block
   From here, send the message to the client.(**Hint:** Use the PrintWriter)

- **ChatClass**

For this class you will have to implement two methods: **open** and **initSenderAndReceiver**

1. In the **open()** method you should make a call to the two other methods

2. In the **initSenderAndReceiver()** you will initialize the receiver
   2.1 The client's socket should be received as the argument in this method
   2.2 Create the receiver using the **Receiver** class
   2.3 Create a thread named **receiverThread** which argument is the created receiver
   2.4 Run the thread

- **SimpleServer**

This class is the main for the server

1. Inside the **main()** create the **Socket clientSocket** and **ServerSocket serverSocket** initialized to null
2. Inside the try-catch block initialize the **serverSocket** to listen to the same port that you used in your Android program
3. Print "Server started. Listening to the port xxxx. Wating to the client" in the console
4. Accept the connection
5. Print "Client connected on port xxxx" in the console.
6. Print a proper message in the catch block.
7. Outside the try-catch block make a call to the ChatClass and initialize it
8. Use the open method to open the connection.

Za FIREWALLS

ZA vo slucaj javna mreza na fakultet command prompt

Note 1: In order the firewalls not to block the ports for using them, you should turn them off.
Note 2: If you are using your own network the connection should be established properly, otherwise do the following steps.

1. Open cmd as Administrator
2. Enter the following to create the connection. Any values for **ssid** and **key** can be given
   netsh wlan set hostednetwork mode=allow ssid=MyWifiName key=myPassword
3. Enter this to start the network
   netsh wlan start hostednetwork
4. **Ensure the wi-fi in on, on your PC**

1.6 References

Good articles about Sockets can be found on the following links provided:

- http://docs.oracle.com/javase/tutorial/networking/sockets/
- http://www.javaworld.com/article/2077322/core-java/core-java-sockets-programming-in-java-a-tutorial.html?null
- http://www.tutorialspoint.com/java/java_networking.htm
- http://developer.android.com/reference/java/net/Socket.html
- http://www.peachpit.com/articles/article.aspx?p=2166868&seqNum=3